REMARKS/ARGUMENTS

Prior to this Amendment, claims 1-24 were pending in the application. Claim 1 is amended to clarify that a condition filter is created in a standard query statement that is precompiled and stored for later repeated use. The condition filter has data values that are unknown at precompiling and uses placeholders for these values that are replaced when the precompiled standard query language statement is used to obtain results from a database. The precompiled statement is then used to respond to additional requests without requiring additional compilation. Dependent claims 2 and 4-8 are amended to provide proper antecedent basis. Claim 3 is canceled.

Independent claim 9 is amended to clarify that the SQL WHERE clause statement can be created and executed to respond to multiple application queries without being recompiled. Further, the method calls for an API to use a tree data structure for storing conditions for data values of unknown parameters, with the data values being replaced by question mark characters during creation of the SQL WHERE clause statement. Claims 10 and 11 are canceled with their limitations being added to claim 9.

Independent claim 12 is amended to clarify the features of the claimed API and to address the rejection of the claim based on indefiniteness.

Independent claim 14 is amended to clarify creation of a condition filter and how a compiled SQL statement with the condition filter can be used to query a database without requiring compilation of the SQL statement containing the filter prior to accessing the database. Dependent claims 15 and 16 are amended to clarify how the condition filter can be created before all values are known and passed by a requesting application. Dependent claims 17-21 are amended to provide correct antecedent basis for each limitation.

Independent claim 22 is amended similarly to claim 14 to clarify how the condition filter is created using a tree data structure to allow unknown parameters to be provided into a precompiled SQL statement, thereby avoiding

the need for recompiling each time the filter is used to query a database. Dependent claims 23 and 24 are canceled.

No new matter is added by this amendment with support found at least in Applicant's specification in paragraphs [0017 – 0018]. Claims 1, 2, 4-9, and 12-22 remain for consideration by the Examiner.

Rejections under 35 U.S.C. 112

In the Office Action, claims 1, 14, and 22 were rejected under 35 U.S.C. 112, second paragraph as being indefinite. Claims 1, 14, and 22 are amended to address unclarity presented with the phrase "using a code to create a condition filter" and with the phrase "sending the standard query language statement to a database." It is believed that after the amendments are made that claims 1, 14, and 22 particularly point out and distinctly claim the subject matter that Applicant regards as the invention.

Additionally, claims 1 and 9 were rejected based on insufficient antecedent basis for one or more limitations. The amendments to the claims address these issues.

Claim Objections

In the Office Action, claims 1-24 were objected to for the use of the abbreviation "JDBC" without the use of parenthesis following descriptive text. The limitations including the "JDBC" abbreviation have been deleted from the claims to address this objection and to clarify the subject matter of the claimed inventions.

Rejections under 35 U.S.C. 103

In the April 8, 2004 Office Action, claims 1-4, 9-17, and 22-24 were rejected under 35 U.S.C. 103 as being unpatentable over U.S. Pat. No. 6,339,768 ("Leung") in view of U.S. Pat. No. 6,356,887 ("Berenson"). This rejection is respectfully traversed based on the amendments to the claims and the following remarks.

The method of claim 1 calls for using a base class to create a condition filter in a standard query language statement. A portion of the data values are replaced with placeholders and a corresponding data value list is created using a tree data structure to store conditions. The method continues with precompiling the standard query language statement including the condition filter and storing the precompiled statement. A query is received from an application that includes values for the data values represented by the placeholders. The precompiled statement is then used to obtain results from a database. The method then includes repeating the query receiving step and the step of using the precompiled statement, whereby the statement is executed multiple times without being recompiled. Because the combination of Leung and Berenson fail to teach or suggest each of these elements, the method of claim 1 is allowable over this combination of references.

Specifically, Leung and Berenson fail to teach precompiling a standard query language statement, storing it for reuse, and then repeatedly using the precompiled statement to request results of a query from a database. In Applicant's October 20, 2003 Amendment, it was clearly explained why Leung fails to teach the execution of a precompiled statement multiple times without requiring it be recompiled. These reasons, which Applicant believes to still be valid, are presented again in the following paragraphs for the sake of clarity and completeness:

"*Leung* has been cited by the Office as describing a precompiled query language statement based on program source code undergoing a "precompile step." *See Leung* col. 5, lines 11–12. While the precompile step in *Leung* may sound similar to precompling a query language statement, it is describing an entirely different event: Nothing is being compiled in the precompile step of *Leung*, but instead, each instance of a query language statement in the source code is replaced with a host language call to the statement, which has been transferred to a separate Database Request Module (DBRM). Only in a later step, after the

9

separated query language statements have been further modified in the DBRM, are they finally compiled.

Each query language statement compiled from the DBRM is mapped to a host call, and there is no suggestion that a host call executes a statement more than once when the source code is executed. In essence, *Leung* is describing the conventional process of compiling and executing query language statements, and does not describe the executing a compiled statement multiple times without the need for recompiling. Furthermore, *Leung* never suggests that precompiling a query language statement and executing it multiple times without recompiling is more efficient than compiling the statement each time it is executed."

Berenson fails to overcome the deficiencies in Leung. Berenson in col. 1, lines 22-56 discusses the use of stored procedures that contain "one or more SQL statements that are compiled once ... and stored in main memory." As discussed above, there is no motivation to modify Leung found in either Leung (which teaches a very different process) or in Berenson (which describes how to make dynamic SQL and not stored procedures except in the Background). Further, even if these references are combined, the combined teaching fails to disclose the method of claim 1 for the following reasons.

Berenson fails to teach the use of a base class to create a condition filter in a standard query language statement. The condition filter includes data values that are replaced by placeholders. The method of claim 1 further calls for data value list corresponding to the data values replaced by the placeholders to be created using a tree data structure to store conditions. Berenson makes a passing reference to using parameters in its stored procedures but fails to teach the creation of the "condition filter" claim 1, i.e., provides no teaching of the use of placeholders, a data value list, and a tree structure for storing conditions.

Further, Berenson fails to teach that its stored procedures are created using a base class, which significantly simplifies construction of the filter and allowing for example, "different SQL WHERE clauses" to be easily formed (see, Applicant's specification at page 20, lines 1-2 and at page 6, para. [0018], which discusses an exemplary base class "CXPFilter" used to construct condition filters according to an embodiment of the invention). For at least these reasons, claim 1 is believed allowable over Leung and Berenson. Claims 2 and 4 depend from claim 1 and are believed allowable as depending from an allowable base claim.

Independent claim 9 is directed to a method of processing a query that includes some limitations similar to that of claim 1, and hence, claim 9 is believed allowable for the reasons provided for allowing claim 1. Further, claim 9 specifically requires the use of an API to create a SQL WHERE clause statement in an SQL statement. The method calls for the SQL WHERE clause to be used to respond to a query from an application such that the SQL WHERE statement is executed multiple times without being recompiled. As discussed with reference to claim 1, Leung completely fails to teach repeatedly using an SQL statement without recompilation, and therefore, also clearly fails to teach doing so with an SQL WHERE clause. Berenson is cited at col. 1, lines 22-38 for providing this teaching, but, at this citation, Berenson is discussing SQL statements in general and does not mention SQL WHERE clauses or indicate how these may be used to avoid recompiling SQL statements when accessing a database. For at least these reasons, the combination of Leung and Berenson fails to teach the method of claim 9.

Further, claim 9 calls for the API to use a tree data structure for storing conditions that is used with the data value list to replace parameters that were unknown at the creation of the SQL WHERE clause with values passed by a requesting application. Neither Leung nor Berenson provide any teaching or suggestion of an API functioning in this manner or the use of tree data structures in combination with data value lists to enable the use of a precompiled SQL WHERE clause. Leung is cited in the Office Action at col. 9, lines 4-15, but the

11

query graph model of Leung does not teach or suggest the use of a tree data structure used with a data value list. For this additional reason, claim 9 is allowable over Leung and Berenson.

Independent claim 12 is believed allowable for the reasons provided for allowing claim 9. Particularly, Leung and Berenson fail to teach creating a condition filter for a SQL WHERE clause. An object is used to request the query such that the SQL WHERE statement can be used multiple times without being recompiled. Claim 13 depends from claims 12 and is believed allowable at least for the reasons for allowing claim 12.

Independent claim 14 is directed to a method with some limitations similar to those of the method of claim 1, and claim 14 is believed allowable for many of the reasons for allowing claim 1. Claims 15-17 depend from claim 14 and are believed allowable as depending from an allowable base claim. Further, claims 15 and 16 include the use of a tree data structure combined with a data value list to allow an application to pass in parameter values and reuse a SQL statement including a condition filter. As noted with reference to claim 9, the Office Action cites Leung for teaching the placeholder and tree data structure features of the invention, but only refers briefly to the model of Figure 9 and related text which fails to teach or suggest these claimed features. Hence, claims 15 and 16 are believed allowable over Leung and Berenson for these additional reasons (and as presented with reference to claim 1).

Independent claim 22 is directed to a computer program product with limitations similar to the combination of claims 14-16, and is believed allowable for the reasons provided for allowing claims 14-16.

Further, in the Office Action, claims 5-8 and 18-21 were rejected under 103(a) as being unpatentable over Leung in view of Berenson further in view of U.S. Pat. No. 5,666,528 ("Thai"). Claims 5-8 and 18-21 depend from claims 1 and 14, respectively, and are believed allowable as depending from allowable base claims. Further, Thai fails to overcome the deficiencies noted above in

12

Leung and Berenson. Specifically, Thai fails to teach a condition filter in a SQL statement that can be used repeatedly to access a database due, at least in part, its unique use of placeholders for unknown data values at time of compilation.

Conclusion

In view of all of the above, the pending claims are now believed to be allowable, and Applicant respectfully requests that a timely Notice of Allowance be issued in this case.

No fee is believed due with this Amendment, but any fee deficiency associated with this submittal may be charged to Deposit Account No. 50-1123.

Respectfully submitted,

June 28, 2004

Kent A. Lembke, Reg. No. 44,866
Hogan & Hartson LLP
One Tabor Center
1200 17th Street, Suite 1500
Denver, Colorado 80202
(720) 406-5378 Tel
(303) 899-7333 Fax

13